

DR



DeepRob

Discussion 2
Introduction to the PROPS Dataset
University of Michigan and University of Minnesota



Today's Agenda

- Administrative Announcements
- Introduction to Project 1
- Discussion of PROPS Dataset
- Troubleshooting



Administrative Announcements

- Project 1 is released
- Autograder roster updated
- Gradescope roster updated
- Slight revision to quiz policy
 - Quizzes to be open all morning on quiz days until start of lecture
 - 7:00AM EST until 3:00PM EST
 - Will be announced again via email



Project 1 – Introduction

- Objective
 - Gain experience building a machine learning pipeline to train and evaluate image classification models
- You will implement three image classifiers
 1. K-Nearest Neighbors
 2. Linear Softmax
 3. Linear SVM



Project 1 – Logistics

- Instructions and code available on the website
 - Here: deeprob.org/projects/project1/
- Uses Python, PyTorch and Google Colab
- Autograder won't be online until this weekend
 - It will be announced once online
- Due Thursday, January 26th 11:59 PM EST





Project 1 – Instructions

The screenshot shows a web browser window for the 'Deep Rob' project. The URL is deeprob.org/projects/project1/. The page title is 'Project 1 | Deep Rob'. The left sidebar has a 'Projects' section with 'Project 1' selected. The main content area shows the 'Project 1' title, 'Overview' (describing the goal of gaining experience building a machine learning pipeline for image classification), 'The goals for this project are as follows:' (a bulleted list of tasks including implementing classifiers and understanding tradeoffs), and 'Instructions' (a numbered list of steps: download starter code, unzip and upload to Google Drive). The University of Michigan Robotics logo is in the top right.

DR Deep Rob

Search Deep Rob

Forum Office Hours Autograder Gradescope

M ROBOTICS
UNIVERSITY OF MICHIGAN

Projects / Project 1

Project 1

Overview

The objective of this project is to gain experience building a machine learning pipeline that can be used to train and evaluate image classification models. In this project you will implement of set of classification models then apply them to a dataset of images in the context of domestic service robots.

The goals for this project are as follows:

- Implement a K-Nearest Neighbors classifier.
- Implement a Multiclass Support Vector Machine classifier.
- Implement a Softmax classifier.
- Understand the differences and tradeoffs between each of these classifiers.
- Understand the characteristics of instance-level classification using the [PROPS dataset](#).
- Practice with cross validating your machine learning models.

Instructions

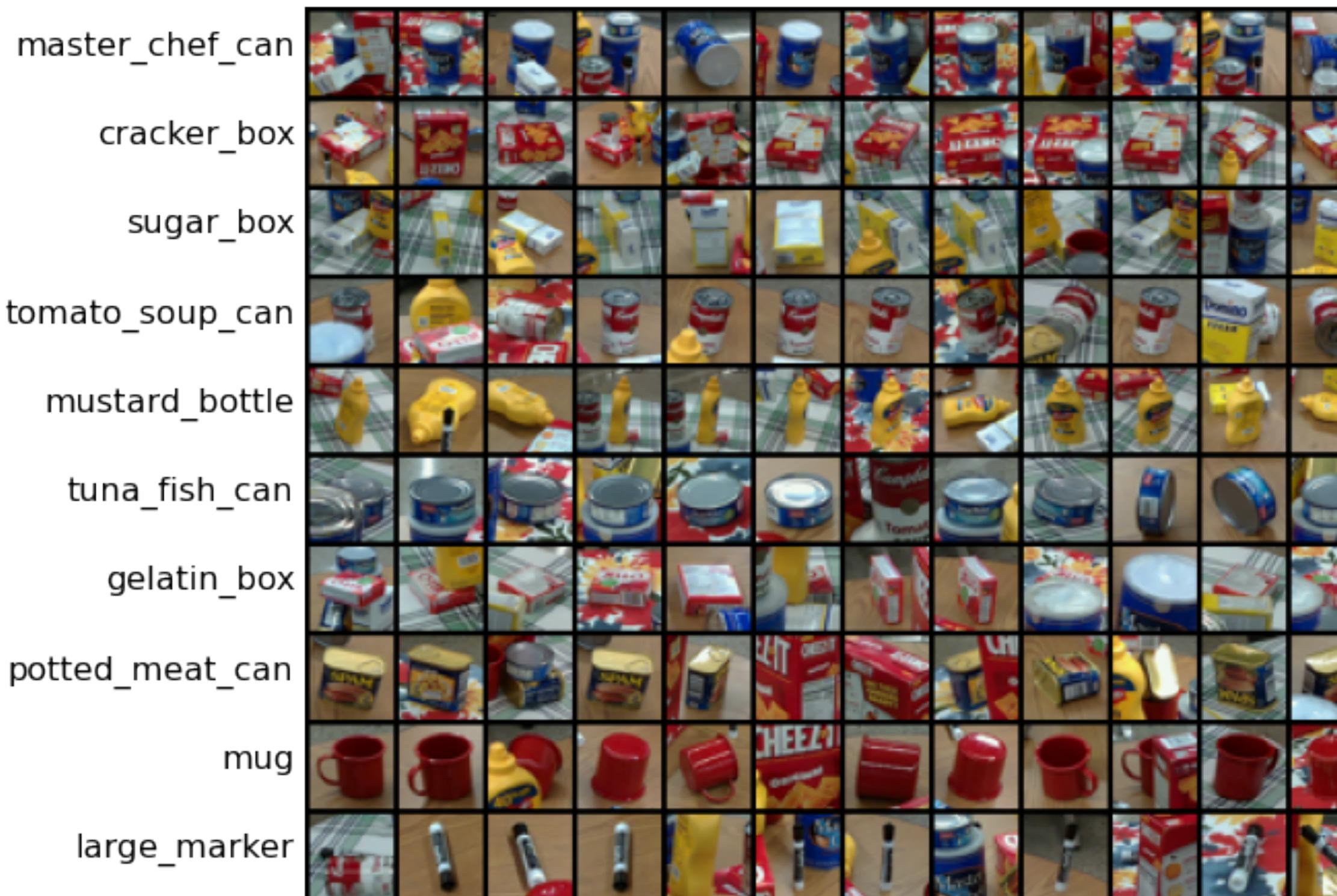
- 1 Download the project starter code
 - [Project 1 starter code: P1.zip](#)
- 2 Unzip the starter code and upload to Google Drive
 - Once unzipped, you should find a root directory titled 'P1'. The 'P1' directory contains all starter code

This site uses [Just the Docs](#), a documentation theme for Jekyll.



Project 1 – Dataset

Progress Robot Object Perception Samples Dataset



10 classes
32x32 RGB images
50k training images (5k per class)
10k test images (1k per class)

Chen et al., “ProgressLabeller: Visual Data Stream Annotation for Training Object-Centric 3D Perception”, IROS, 2022.



Project 1 — Setup & Preprocessing

The screenshot shows a Google Colab notebook titled "knn.ipynb". The left sidebar contains a "Table of contents" with sections like "ROB 498-002/599-009 Project 1-1: K-Nearest Neighbors (k-NN)", "Setup Code", "Google Colab Setup", "Data preprocessing / Visualization", "Load the Progress Objects Dataset", "Visualize the dataset", and "Subsample the dataset". The main content area is titled "Load the Progress Objects Dataset" and describes the utility function `rob599.data.progress_objects()`. It explains that the function returns a sample of 10 objects from the [Progress Objects dataset](#) as a set of four **Torch tensors**: `x_train`, `y_train`, `x_test`, and `y_test`. A bulleted list details the contents of each tensor. Below this, a note states that the function automatically downloads the UW Object dataset the first time it's run. A code cell at the bottom shows the command being run:

```
[6] x_train, y_train, x_test, y_test = rob599.data.progress_objects()  
20s  
print('Training set:', )  
print(' data shape:', x_train.shape)  
print(' labels shape: ', y_train.shape)  
print('Test set:')  
print(' data shape: ', x_test.shape)  
print(' labels shape', y_test.shape)  
  
Downloading https://topipari.com/data/Progress-Objects-Sample.tar.gz to ./Prog  
100% 171571589/171571589 [00:15<00:00,  
16018662.57it/s]  
  
Extracting ./Progress-Objects-Sample.tar.gz to .  
Training set:  
 data shape: torch.Size([50000, 3, 32, 32])  
 labels shape: torch.Size([50000])  
  
1s completed at 2:40 PM
```



Project 1 — Setup & Preprocessing

The screenshot shows a Google Colab notebook titled "knn.ipynb". The left sidebar contains a "Table of contents" with sections like "ROB 498-002/599-009 Project 1-1: K-Nearest Neighbors (k-NN)", "Setup Code", "Google Colab Setup", "Data preprocessing / Visualization", "Load the Progress Objects Dataset", "Visualize the dataset", and "Subsample the dataset". The main area shows code to load the dataset:

```
[6] x_train, y_train, x_test, y_test = rob599.data.progress_objects()  
  
print('Training set:', )  
print(' data shape:', x_train.shape)  
print(' labels shape: ', y_train.shape)  
print('Test set: ')  
print(' data shape: ', x_test.shape)  
print(' labels shape', y_test.shape)  
  
Downloading https://topipari.com/data/Progress-Objects-Sample.tar.gz to ./Progress-Objects-Sample  
100% [██████████] 171571589/171571589 [00:15<00:00,  
16018662.57it/s]  
  
Extracting ./Progress-Objects-Sample.tar.gz to .  
Training set:  
 data shape: torch.Size([50000, 3, 32, 32])  
 labels shape: torch.Size([50000])  
  
1s completed at 2:40 PM
```

```
Extracting ./Progress-Objects-Sample.tar.gz to .  
Training set:  
 data shape: torch.Size([50000, 3, 32, 32])  
 labels shape: torch.Size([50000])  
Test set:  
 data shape: torch.Size([10000, 3, 32, 32])  
 labels shape torch.Size([10000])
```

Verify size of train, test set





Project 1 – Dataset Visualization

The screenshot shows a Google Colab notebook titled "knn.ipynb" with the URL colab.research.google.com/drive/1zpeAd2QC7otqVbRuReXLwM-zMQHKjyY2#scrollTo=-nLyYUhBgDKp. The notebook has a "Table of contents" sidebar on the left and a main workspace on the right.

In the workspace, under the heading "Visualize the dataset", there is a text block explaining the purpose of the visualization:

To give you a sense of the nature of the images in the Progress Objects dataset, this cell visualizes some random examples from the training set.

Below the text is a code cell (cell 7) containing Python code to generate a grid of images. The code imports `random` and `torchvision.utils`, defines a list of class names, and uses `plt.text` and `plt.imshow` to display a grid of images for three classes: "master_chef_can", "cracker_box", and "sugar_box".

```
[7]: import random
from torchvision.utils import make_grid

classes = [
    "master_chef_can",
    "cracker_box",
    "sugar_box",
    "tomato_soup_can",
    "mustard_bottle",
    "tuna_fish_can",
    "gelatin_box",
    "potted_meat_can",
    "mug",
    "large_marker"
]

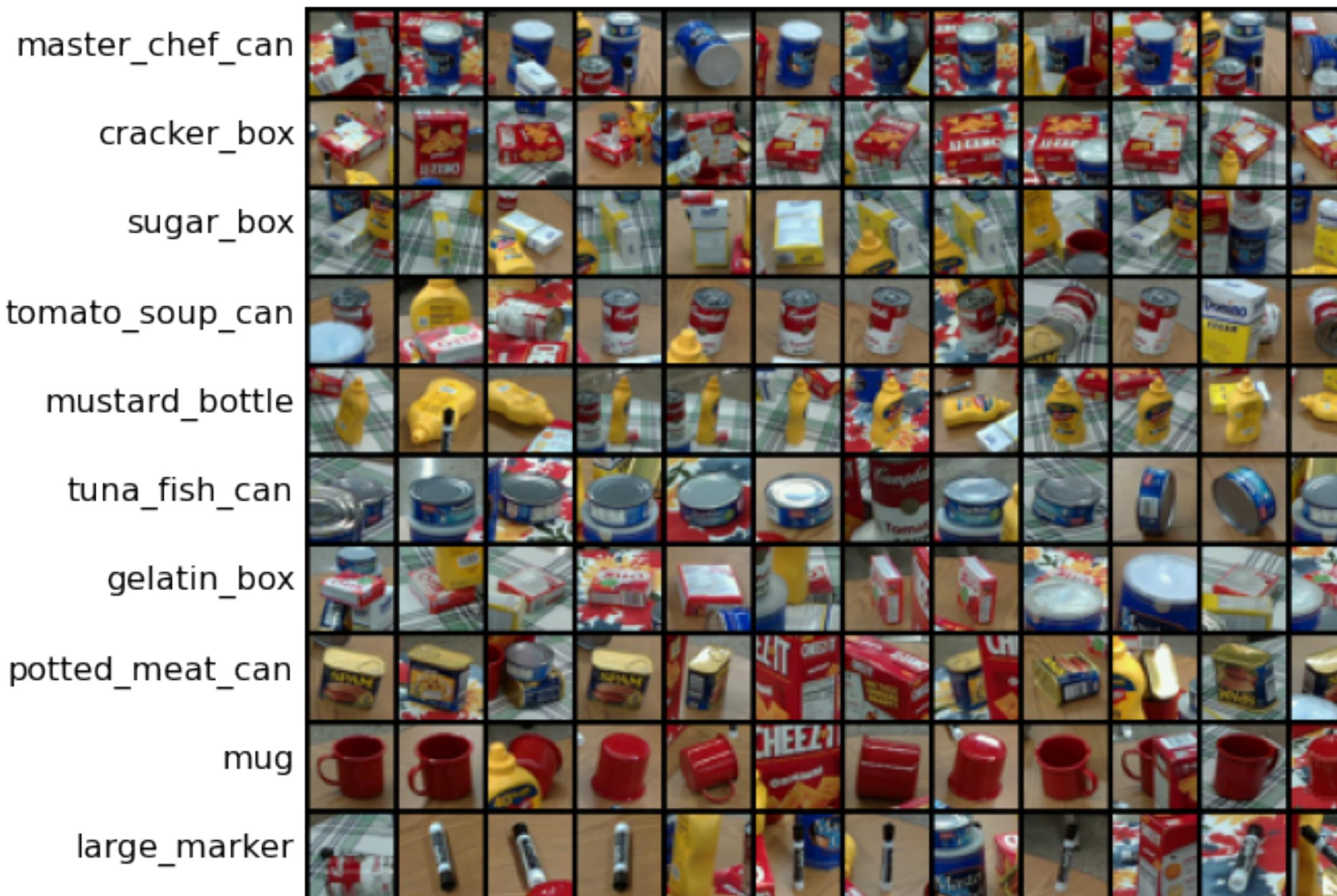
samples_per_class = 12
samples = []
for y, cls in enumerate(classes):
    plt.text(-4, 34 * y + 18, cls, ha='right')
    idxs, = (y_train == y).nonzero(as_tuple=True)
    for i in range(samples_per_class):
        idx = idxs[random.randrange(idxs.shape[0])].item()
        samples.append(x_train[idx])
img = torchvision.utils.make_grid(samples, nrow=samples_per_class)
plt.imshow(rob599.tensor_to_image(img))
plt.axis('off')
plt.show()
```

The resulting visualization shows a 3x12 grid of images for each class. The first row contains images of "master_chef_can", the second row contains images of "cracker_box", and the third row contains images of "sugar_box".



Project 1 – Dataset

Progress Robot Object Perception Samples Dataset



10 classes
32x32 RGB images
50k training images (5k per class)
10k test images (1k per class)

Chen et al., “ProgressLabeller: Visual Data Stream Annotation for Training Object-Centric 3D Perception”, IROS, 2022.

Project 1 – Dataset

Progress Robot Object Perception Samples Dataset

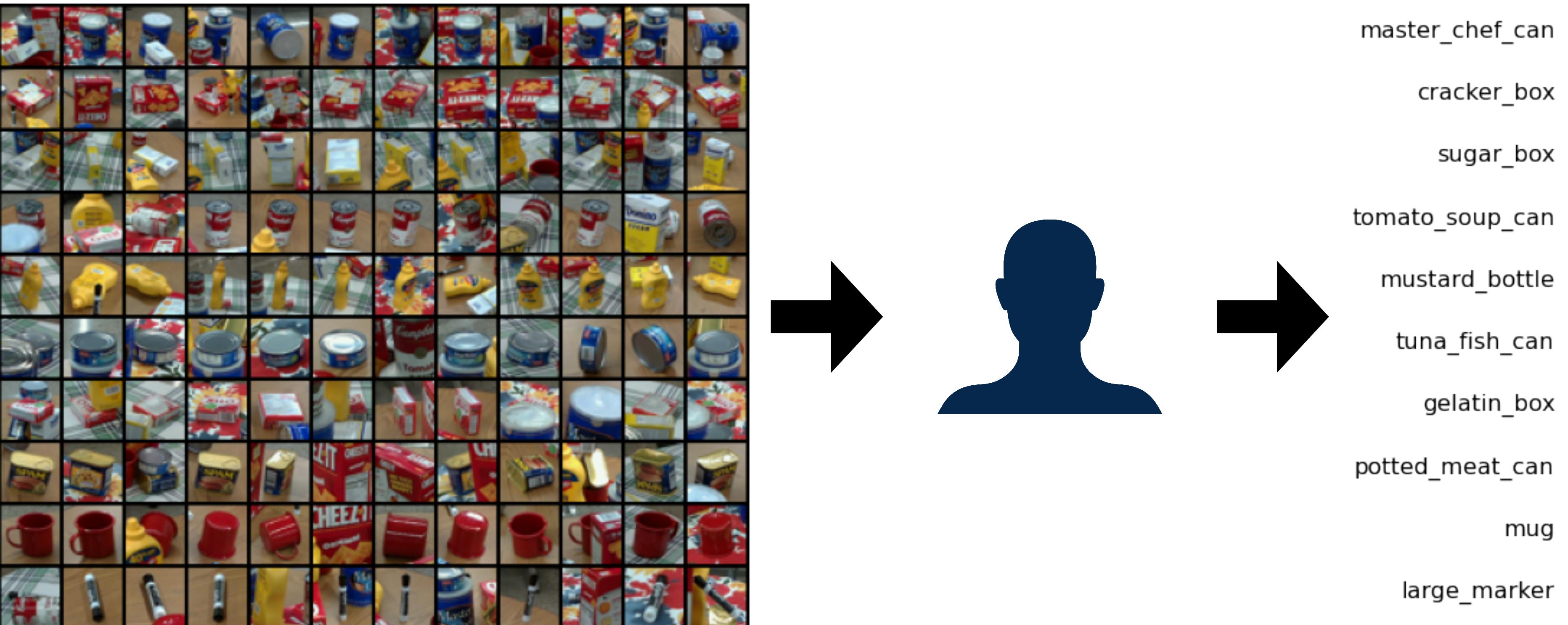


10 classes
32x32 RGB images
50k training images (5k per class)
10k test images (1k per class)

What is the source of this data?

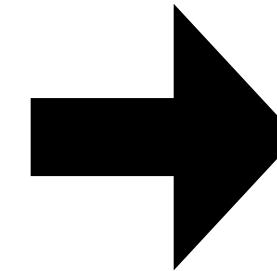
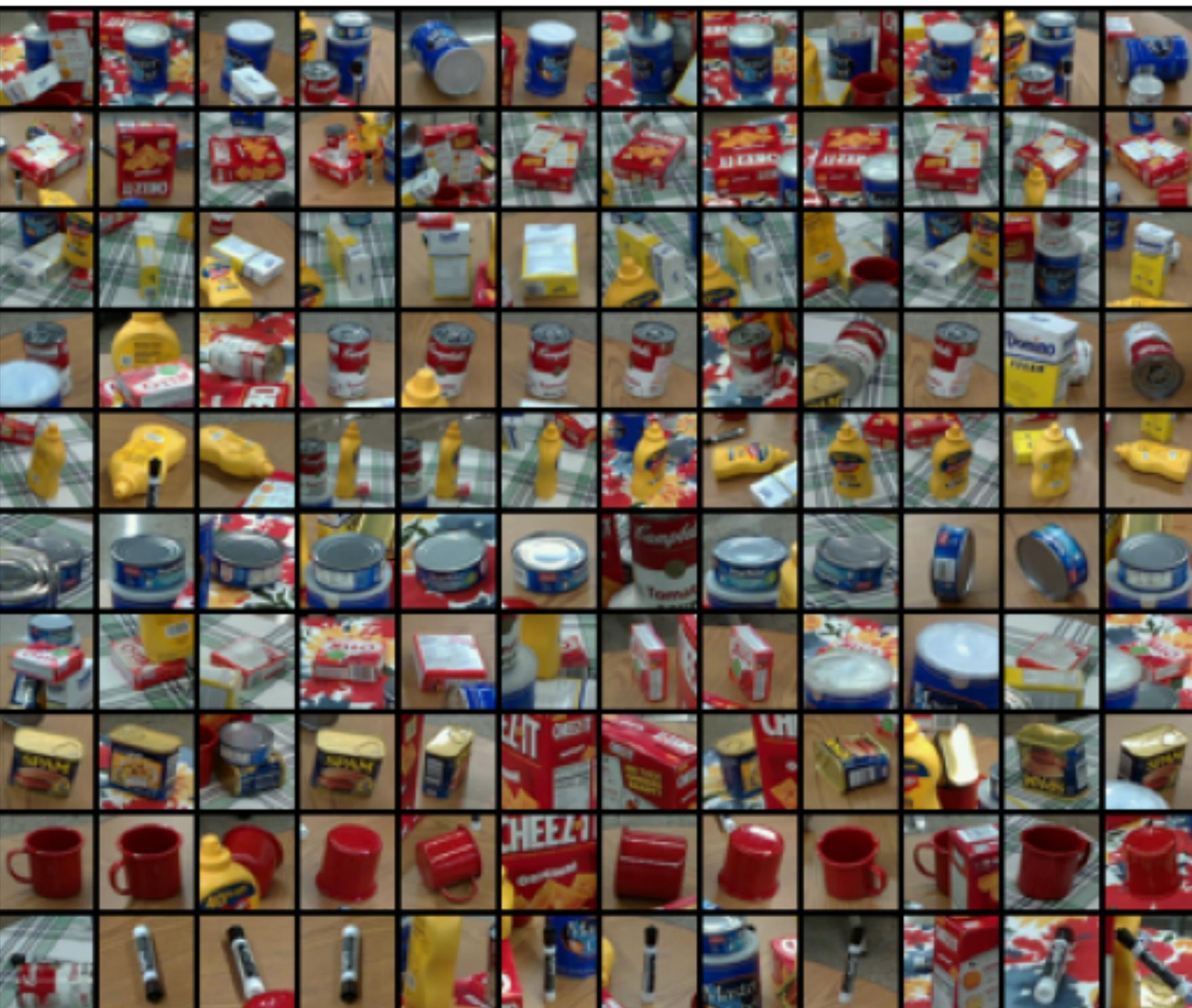
Chen et al., “ProgressLabeller: Visual Data Stream Annotation for Training Object-Centric 3D Perception”, IROS, 2022.

Labeling a Dataset

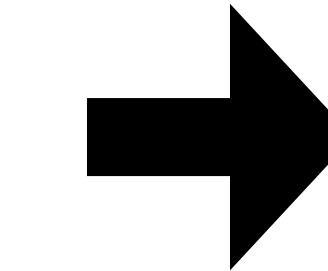


Chen et al., "ProgressLabeller: Visual Data Stream Annotation for Training Object-Centric 3D Perception", IROS, 2022.

Labeling a Dataset



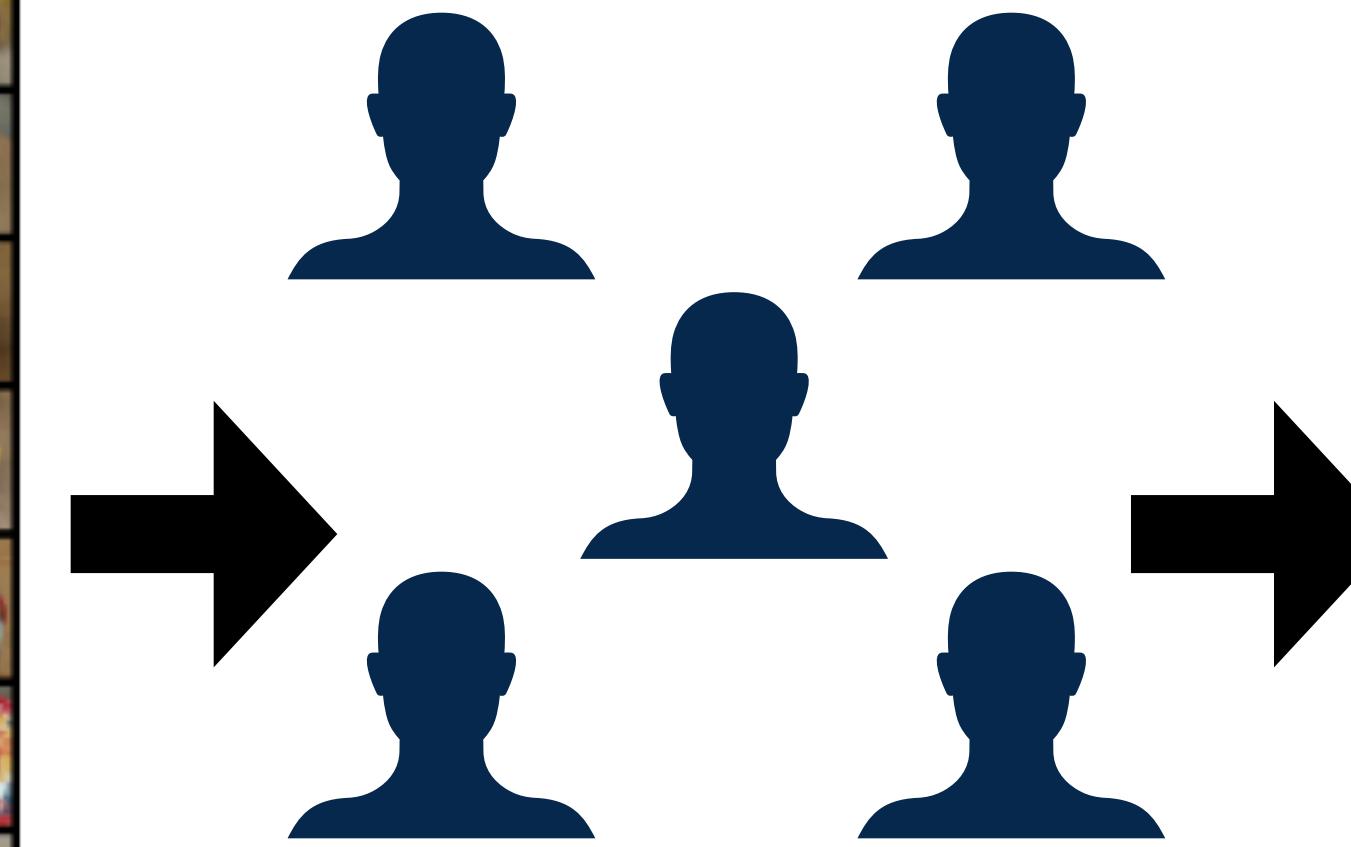
**Human
Annotator**



master_chef_can
cracker_box
sugar_box
tomato_soup_can
mustard_bottle
tuna_fish_can
gelatin_box
potted_meat_can
mug
large_marker

Chen et al., "ProgressLabeller: Visual Data Stream Annotation for Training Object-Centric 3D Perception", IROS, 2022.

Labeling a Dataset

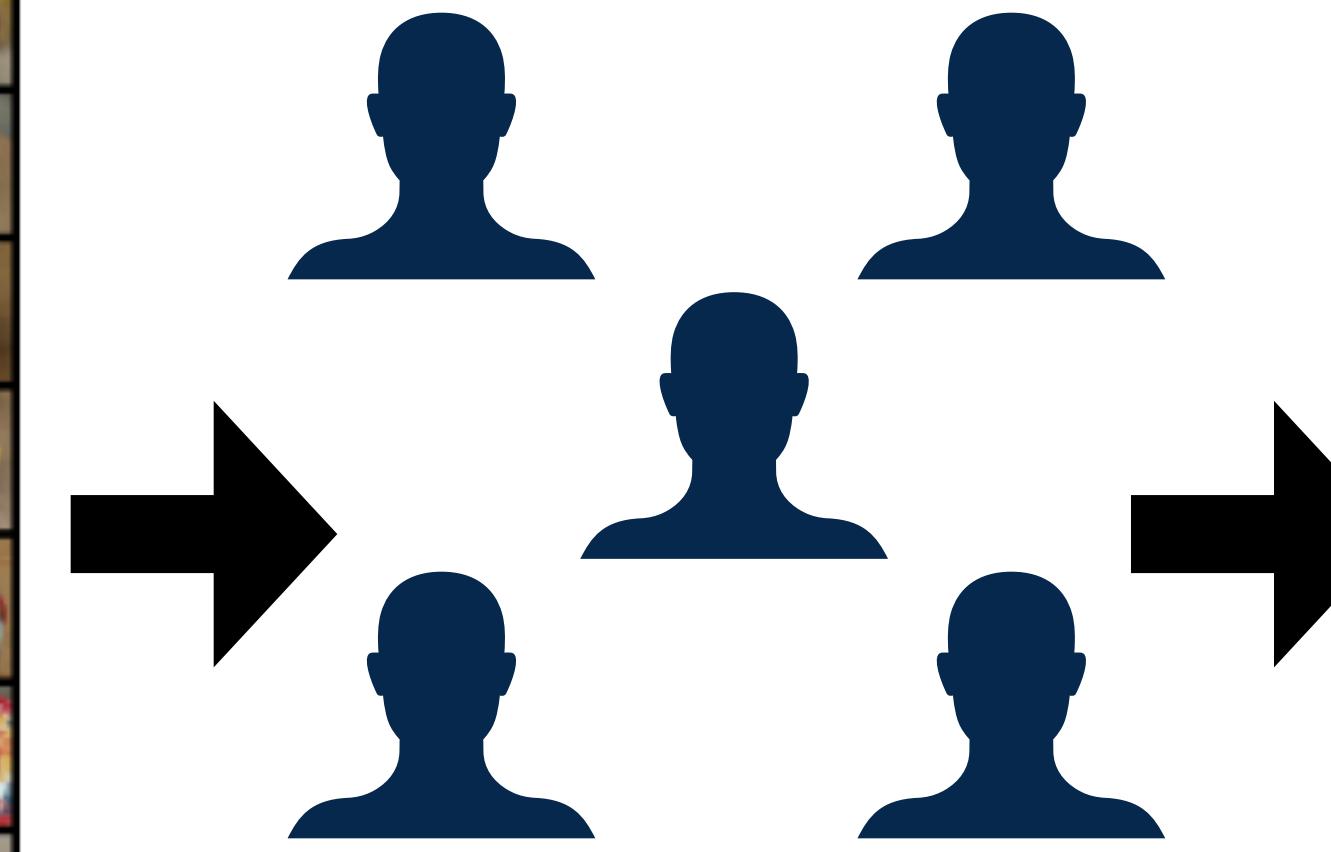


master_chef_can
cracker_box
sugar_box
tomato_soup_can
mustard_bottle
tuna_fish_can
gelatin_box
potted_meat_can
mug
large_marker

Chen et al., "ProgressLabeller: Visual Data Stream Annotation for Training Object-Centric 3D Perception", IROS, 2022.

Labels are expensive!

Labeling a Dataset



**Human
Annotators**

master_chef_can
cracker_box
sugar_box
tomato_soup_can
mustard_bottle
tuna_fish_can
gelatin_box
potted_meat_can
mug
large_marker

Chen et al., "ProgressLabeller: Visual Data Stream Annotation for Training Object-Centric 3D Perception", IROS, 2022.

Labels are expensive!

ProgressLabeller—A Tool to Annotate 3D Objects

**ProgressLabeller: Visual Data Stream Annotation for Training
Object-Centric 3D Perception**

Xiaotong Chen Huijie Zhang Zeren Yu Stanley Lewis Odest Chadwicke Jenkins

Abstract— Visual perception tasks often require vast amounts of labelled data, including 3D poses and image space segmentation masks. The process of creating such training data sets can prove difficult or time-intensive to scale up to efficacy for general use. Consider the task of pose estimation for rigid objects. Deep neural network based approaches have shown good performance when trained on large, public datasets. However, adapting these networks for other novel objects, or fine-tuning existing models for different environments, requires significant time investment to generate newly labelled instances. Towards this end, we propose ProgressLabeller as a method for more efficiently generating large amounts of 6D pose training data from color images sequences for custom scenes in a scalable manner. ProgressLabeller is intended to also support transparent or translucent objects, for which the previous methods based on depth dense reconstruction will fail. We demonstrate the effectiveness of ProgressLabeller by rapidly create a dataset of over 1M samples with which we fine-tune a state-of-the-art pose estimation network in order to markedly improve the downstream robotic grasp success rates. ProgressLabeller is open-source at <https://github.com/huijieZH/ProgressLabeller>

I. INTRODUCTION

Fig. 1: The ProgressLabeller offers an interactive GUI for aligning all kinds of objects in the 3D scene to generate large-scale datasets with ground truth pose labels. The left image shows the rough 6D pose estimates from one state-of-the-art RGB-D deep models trained on public YCB dataset, and the right images shows fine-tuned pose estimates from the same model after retraining using generated data from ProgressLabeller. The pose estimates are then used for robotic grasping experiments.

**Reduces the time needed to generate labels
for object classification, pose, segmentation**



ProgressLabeller—A Tool to Annotate 3D Objects

**Rough Pose Estimates
from Pretrained Model**



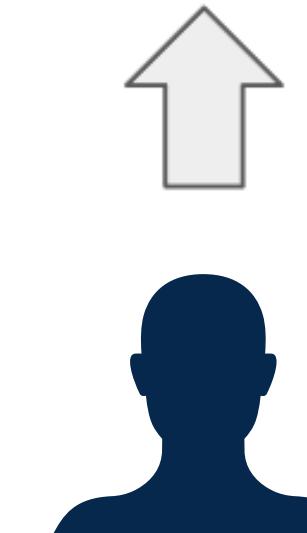
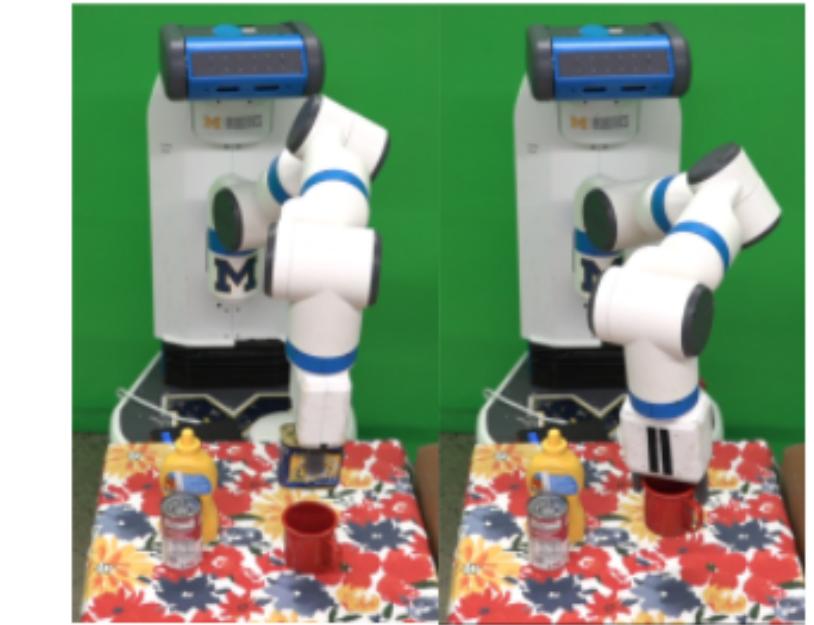
**6D pose annotation through
interactive interface**



**Fine-tuned Pose
Estimates**



**Pose-based Robot
Grasping**



Idea:

1. Record video of scene
2. Human labels object pose in selected frames
3. Pose labels propagate to (large number of) remaining frames

ProgressLabeller—A Tool to Annotate 3D Objects

**Rough Pose Estimates
from Pretrained Model**



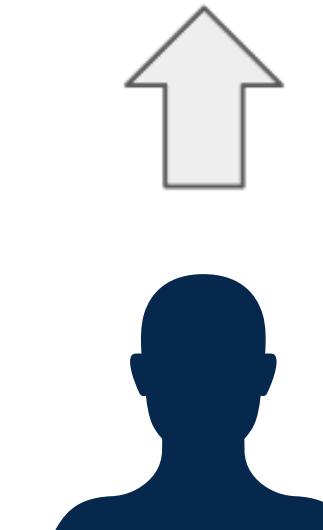
**6D pose annotation through
interactive interface**



**Fine-tuned Pose
Estimates**



**Pose-based Robot
Grasping**



**Human
Annotator**

Idea:

1. Record video of scene
2. Human labels object pose in selected frames
3. Pose labels propagate to (large number of) remaining frames

ProgressLabeller—A Tool to Annotate 3D Objects



ProgressLabeller—A Tool to Annotate 3D Objects

